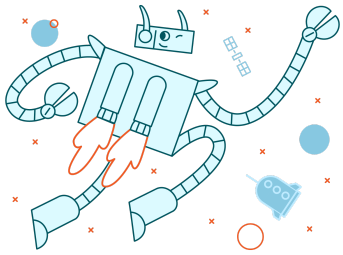
Strings Contd

With strings, there is a lot more we can do aside from assigning strings and concatenating them. We will learn more about strings in this lesson.



Strings

Previously, we learnt that you can represent text in JavaScript using strings. String are surrounded by quotes `'` or double quotes `"`.

```
console.log("I love JavaScript!")
```

Console:

```
I love JavaScript!
```

And you can concatenate strings using the `+` operator. Concatenation appends one string to the end of another.

```
console.log("I know " + "HTML and JavaScript!")
```

Console:

```
I know HTML and JavaScript!
```

New Line

With strings (especially looong ones), we may want to represent part of the string in an actual new-line.

We can represent a line-break in our string by adding `\n` before each new line starts.

```
let myString = "You are not a drop in the ocean.\n"
+ "You are the entire ocean, in a drop."
```

```
"You are not a drop in the ocean.
You are the entire ocean, in a drop."
```

Representing `''` as a string

`''` is used to represent strings. `\` allows us to represent `''` as a **character in our strings**.

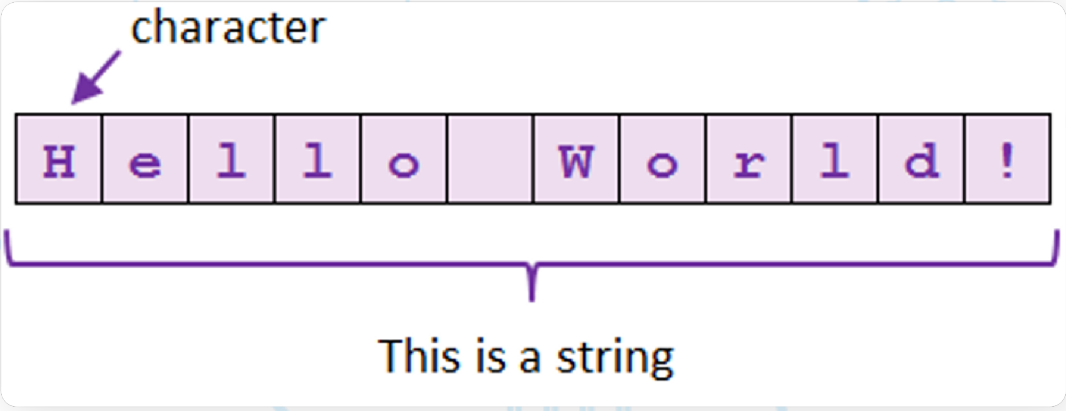
```
let myString = "\"hello\""
```

Console:

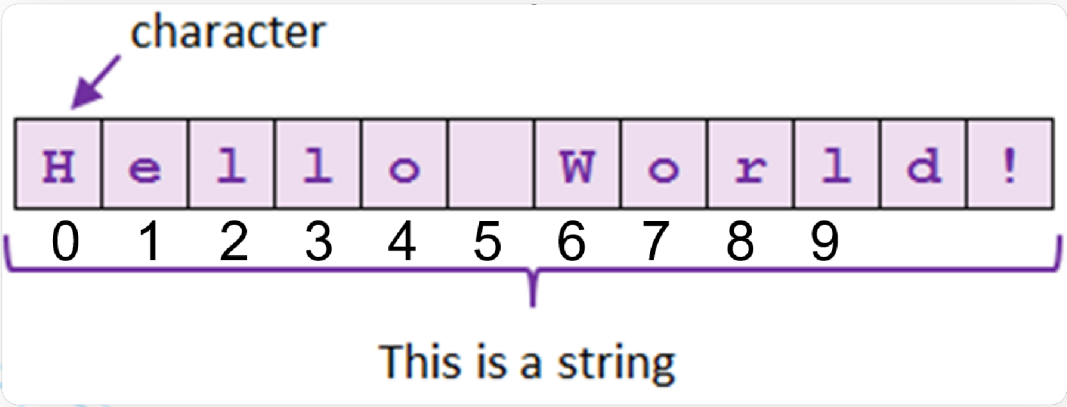
```
""hello""
```

Index of Characters

Strings are just a **bunch of characters** "stringed" together in a series. It is possible for us to get a particular character in a string.



The first character in the string has index 0. This is because we start counting from 0 instead of 1.



To get a particular character, we can add `[]` to the back of the variable/string. We put the "index" between the `[]` brackets.

```
let myString = "Hello World!"
let firstChar = myString[0]
let sixthChar = myString[5]
console.log(firstChar)
console.log(sixthChar)
```

Console:

```
"H"
" "
```

Note: A space counts as a character too!

Because we start counting from 0, the last character of our string will always have the index `string.length - 1`.

```
let myString = "my string"
let lastChar = myString[myString.length-1]
console.log(lastChar)
```

Console:

```
"g"
```

Length of a string

Every string has a **"length"** property with the number of characters in the string. We can get the length of a string by adding `.length` to the back of your variable/string.

```
let myString = "Hello World!"
console.log(myString.length)
```

Console:

```
12
```

Converting strings

JavaScript provides two helpful methods to convert text.

to upper

`.toUpperCase()` can be used to convert **all** the characters in your string to upper case.

```
let myMessage = "Pineapple 1 dollar!".toUpperCase()
console.log(myMessage)
```

Console:

```
"PINEAPPLE 1 DOLLAR!"
```

Note: Characters that are already uppercase remain unchanged. If the character is not a letter, it remains the same.

to lower

`.toLowerCase()` can be used to convert all the characters in your string to lower case.

```
let myMessage = "PINEAPPLE 1 DOLLAR!".toLowerCase()
console.log(myMessage)
```

Console:

```
"pineapple 1 dollar!"
```

Note: Characters that are already lowercase remain unchanged. If the character is not a letter, it remains the same.

Replace occurrences of a substring

`.replace(original phrase, new phrase)` will replace the original phrase in the string with the new phrase.

Notice that `.replace()` doesn't modify the string, it "returns" a new string after having performed the substitution. That's why we need to assign the value of message to replace the message every time.

```
let message  = "send the money to the drop-off
tomorrow at 1"

message = message.replace("money", "tea")
message = message.replace("dropoff", "river")
message = message.replace("send", "throw")

console.log(message)
```

`.replace()` will only replace the first instance of the original phrase. As such to replace all instances, we can use `.replaceAll()`.

```
let message = "Please bring some coffee to the
coffee shop today"
let newMessage = message.replaceAll("coffee", "tea")

console.log(newMessage)
```

Console:

```
"Please bring some tea to the tea shop today"
```

Return char between starting and ending index

"Slicing" returns the characters between the starting index and the ending index (end not included). We can do so by by adding `.slice(starting index, ending index)` to the back of your variable/string.

```
let myString = "Hello Kass!"
myString = myString.slice(6,10)
console.log(myString)
```

Console:

```
Kass
```

Note: The ending index is not included! For example, (2, 4) will return the characters from indexes 2 and 3. Ensure that you have counted carefully.

Additional Readings

Template Strings

Template strings provides us an easy way to insert variables and expressions into a string.

```
let a = 1337
let myString = `The number is ${a}`
console.log(myString)
```

Console:

```
"The number is 1337"
```

The `${}` is used to reference a variable within a string. This can be especially helpful when you have to insert a lot of variables or expressions into a string.

```
var item1 = "steak"
var item2 = "potatoes"
var item3 = "bubble tea"

let templateString = `Today i ate ${item1} with
${item2}. I had ${item3} for dessert.`
let normalString = "Today I ate " + item1 + " with "
+ item2 + "." + " I had " + item3 + " for dessert."

console.log(templateString)
console.log(normalString)
```

Console:

```
"Today I ate steak with potatoes. I had bubble tea
for dessert."
"Today I ate steak with potatoes. I had bubble tea
for dessert."
```

Pay attention that you must use backticks `` ` `` in order to define a template string. It won't work with regular quotes.